Problem1

Code:

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score

X_train, X_test, y_train, y_test = train_test_split(df.iloc[:, :-1], df['target'], test_size=0.3, random_state=42)

for depth in range(1, 6):
    clf = DecisionTreeClassifier(max_depth=depth, min_samples_leaf=2, min_samples_split=5, random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    print(f"Depth {depth}:")
    print(f"  Recall: {recall_score(y_test, y_pred, average='macro')}")
    print(f"  Precision: {precision_score(y_test, y_pred, average='macro')}")
    print(f"  F1 Score: {f1_score(y_test, y_pred, average='macro')}")
```

Result:

Depth 1:
  Recall: 0.6666666666666666
  Precision: 0.5
  F1 Score: 0.5555555555555555
Depth 2:
  Recall: 0.9743589743589745
  Precision: 0.9761904761904763
  F1 Score: 0.974320987654321
Depth 3:
  Recall: 1.0
  Precision: 1.0
  F1 Score: 1.0
Depth 4:
  Recall: 1.0
  Precision: 1.0
  F1 Score: 1.0

Depth 5:
Recall: 1.0
Precision: 1.0
F1 Score: 1.0

1. Which depth values result in the highest Recall? Why?
   When Depth = 3, 4, and 5, Recall reaches 1.0, which means that all positive samples are correctly classified.
   The reason is that the tree is deep enough to capture all patterns in the data.
2. Which value resulted in the lowest Precision? Why?
   When Depth = 1, Precision is the lowest (0.5).
   The reasons are as follows
   First, the tree is too shallow to correctly distinguish the 3 types of data, and can only be classified into 2 categories, resulting in more classification errors.
   Second, some categories may not be predicted at all, resulting in Precision becoming 0.
3. Which value results in the best F1 score? Also, explain the difference between the micro, macro, and weighted methods of score calculation
   If we only look at the F1 score, the best depth is 3 and above, as it reaches 1.0.
   If we consider generalization ability, depth 2 may be a more reasonable choice, as it is almost perfect, but may not overfit as much as depth 3 and above.

Problem2

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier

# Load data
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data"
df = pd.read_csv(url, header=None, na_values="?")
df.dropna(inplace=True)
df = df.astype(int)

# Define features and target
df.columns = ['ID', 'Clump_Thickness', 'Uniformity_Cell_Size', 'Uniformity_Cell_Shape', 'Marginal_Adhesion',
              'Single_Epithelial_Cell_Size', 'Bare_Nuclei', 'Bland_Chromatin', 'Normal_Nucleoli', 'Mitoses', 'Class']
X = df.iloc[:, 1:-1]
y = df['Class']

# Train decision tree
```

```
clf = DecisionTreeClassifier(max_depth=2, min_samples_leaf=2, min_samples_split=5, criterion='gini', random_state=42)
clf.fit(X, y)

# Compute Gini, Entropy, and Misclassification Error
def calculate_metrics(tree):
    root = tree.tree_
    gini = root.impurity[0]
    entropy = -(gini * np.log2(gini + 1e-9) + (1 - gini) * np.log2(1 - gini + 1e-9))
    misclassification = 1 - np.max(root.value[0]) / np.sum(root.value[0])
    return gini, entropy, float(misclassification)

gini, entropy, misclassification = calculate_metrics(clf)
print(f"Gini: {gini:.4f}, Entropy: {entropy:.4f}, Misclassification Error: {misclassification:.4f}")

# Compute Information Gain
info_gain = entropy - np.sum(clf.tree_.weighted_n_node_samples[1:] / clf.tree_.weighted_n_node_samples[0] * clf.tree_.impurity[1:])
print(f"Information Gain: {info_gain:.4f}")

# Get first split feature and threshold
feature_index = clf.tree_.feature[0]
feature_name = X.columns[feature_index]
threshold = clf.tree_.threshold[0]

print(f"First split feature: {feature_name}")
print(f"Decision boundary value: {threshold:.2f}")
```

Result：

Gini: 0.4550, Entropy: 0.9941, Misclassification Error: 0.3499
Information Gain: 0.7825
First split feature: Uniformity_Cell_Size
Decision boundary value: 2.50

1. Calculate the Entropy, Gini, and Misclassification Error of the first split.
   Gini: 0.4550, Entropy: 0.9941, Misclassification Error: 0.3499
2. What is the Information Gain?
   Information Gain: 0.7825
3. Which feature is selected for the first split, and what value determines the decision boundary?
   First split feature: Uniformity_Cell_Size, Decision boundary value: 2.50

Problem 3

```
import pandas as pd
import numpy as np
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"
df = pd.read_csv(url, header=None)

# Define features and target
X = df.iloc[:, 2:].values
y = df.iloc[:, 1].map({'M': 1, 'B': 0}).values  # Convert labels to binary (M = 1, B = 0)

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

# Train decision tree on original data
clf_original = DecisionTreeClassifier(max_depth=2, min_samples_leaf=2, min_samples_split=5, criterion='gini', random_state=42)
clf_original.fit(X_train, y_train)
y_pred_original = clf_original.predict(X_test)

# Compute evaluation metrics for original data
precision_original = precision_score(y_test, y_pred_original)
recall_original = recall_score(y_test, y_pred_original)
f1_original = f1_score(y_test, y_pred_original)

# Apply PCA (1 component)
pca = PCA(n_components=1)
X_train_pca1 = pca.fit_transform(X_train)
X_test_pca1 = pca.transform(X_test)

# Train decision tree on first principal component
clf_pca1 = DecisionTreeClassifier(max_depth=2, min_samples_leaf=2, min_samples_split=5, criterion='gini', random_state=42)
clf_pca1.fit(X_train_pca1, y_train)
```

```
y_pred_pca1 = clf_pca1.predict(X_test_pca1)

# Compute evaluation metrics for PCA (1 component)
precision_pca1 = precision_score(y_test, y_pred_pca1)
recall_pca1 = recall_score(y_test, y_pred_pca1)
f1_pca1 = f1_score(y_test, y_pred_pca1)

# Apply PCA (2 components)
pca = PCA(n_components=2)
X_train_pca2 = pca.fit_transform(X_train)
X_test_pca2 = pca.transform(X_test)

# Train decision tree on first two principal components
clf_pca2 = DecisionTreeClassifier(max_depth=2, min_samples_leaf=2,
min_samples_split=5, criterion='gini', random_state=42)
clf_pca2.fit(X_train_pca2, y_train)
y_pred_pca2 = clf_pca2.predict(X_test_pca2)

# Compute evaluation metrics for PCA (2 components)
precision_pca2 = precision_score(y_test, y_pred_pca2)
recall_pca2 = recall_score(y_test, y_pred_pca2)
f1_pca2 = f1_score(y_test, y_pred_pca2)

# Compute confusion matrix for PCA (1 component)
cm_pca1 = confusion_matrix(y_test, y_pred_pca1)
TP_pca1 = cm_pca1[1, 1]
FP_pca1 = cm_pca1[0, 1]
FN_pca1 = cm_pca1[1, 0]
TN_pca1 = cm_pca1[0, 0]

FPR_pca1 = FP_pca1 / (FP_pca1 + TN_pca1)
TPR_pca1 = TP_pca1 / (TP_pca1 + FN_pca1)

print("Original Data Performance:")
print(f"Precision: {precision_original:.4f}, Recall: {recall_original:.4f}, F1 Score:
{f1_original:.4f}\n")

print("PCA (1 Component) Performance:")
  print(f"Precision: {precision_pca1:.4f}, Recall: {recall_pca1:.4f}, F1 Score: {f1_pca1:.4f}")
print(f"False Positives: {FP_pca1}, True Positives: {TP_pca1}")
print(f"False Positive Rate: {FPR_pca1:.4f}, True Positive Rate: {TPR_pca1:.4f}\n")

print("PCA (2 Components) Performance:")
print(f"Precision: {precision_pca2:.4f}, Recall: {recall_pca2:.4f}, F1 Score: {f1_pca2:.4f}")
```

Result:
Original Data Performance:
Precision: 0.9048, Recall: 0.9048, F1 Score: 0.9048

PCA (1 Component) Performance:
Precision: 0.8923, Recall: 0.9206, F1 Score: 0.9062
False Positives: 7, True Positives: 58
False Positive Rate: 0.0648, True Positive Rate: 0.9206

PCA (2 Components) Performance:
Precision: 0.9310, Recall: 0.8571, F1 Score: 0.8926

1. Using only the first principal component of the data for model fitting, what are the F1 score, Precision, and Recall of the PCA-based single factor model compared to the original (continuous) data?

original data:	PCA-based single factor model:
Precision: 0.9048	Precision: 0.8923
Recall: 0.9048	Recall: 0.9206
F1 Score: 0.9048	F1 Score: 0.9062

2. Repeat the process using the first and second principal components. Using the Confusion Matrix, what are the values for False Positives (FP) and True Positives (TP), as well as the False Positive Rate (FPR) and True Positive Rate (TPR)?
   False Positives（FP）: 7
   True Positives（TP）: 58
   False Positive Rate（FPR）: 0.0648
   True Positive Rate（TPR）: 0.9206
3. Is using continuous data beneficial for the model in this case? How?
   Using continuous data is more beneficial to the model.

   Since the F1 Score of the original data is higher than that of the data processed by PCA, it shows that the complete feature information is helpful for classification. Although PCA-based single factor model is close to the performance of the original data, the information loss, especially the decrease in precision, affects the accuracy of the model.
   In addition, PCA may reduce the computational complexity, but in this case, retaining all continuous features provides more complete classification information and improves the overall performance.